Using Decision-Tree Classifier Systems to Extract Knowledge from Databases*

D.C. St. Clair ,
C. L. Sabharwal
University of MO--Rolla
Graduate Engineering Center
St. Louis, MO 63121

Keith Hacke,
W.E. Bond
McDonnell Douglas Research Laboratories
St. Louis, MO 63166

ABSTRACT

One difficulty in applying artificial intelligence techniques to the solution of "real world" problems is that the development and maintenance of many AI systems, such as those used in diagnostics, require large amounts of human resources. At the same time, databases frequently exist which contain information about the process(es) of interest. Recently, efforts to reduce development and maintenance costs of AI systems have focused on using machine learning techniques to extract knowledge from existing databases. This paper describes research conducted at McDonnell Douglas Research Laboratories in the area of knowledge extraction using a class of machine learning techniques called decision-tree classifier systems. Results of this research suggest ways of performing knowledge extraction which may be applied in numerous situations. In addition, a measurement called the Concept Strength Metric (CSM) is described which can be used to determine how well the resulting decision tree can differentiate between the concepts it has learned. The CSM can be used to determine whether or not additional knowledge needs to be extracted from the database. An experiment involving "real world" data is presented to illustrate the concepts described.

INTRODUCTION

Applying AI techniques to solve diagnostic problems often requires that information contained in one or more databases be converted to knowledge. One common way of performing this conversion is to use domain experts. For example, when experts are asked to assemble a set of rules for diagnosing a particular system, they review information from sources such as schematics and existing maintenance databases. Then they develop a set of diagnostic concepts, generally stated as a set of rules, which correlate diagnostic inputs with the desired diagnosis(es). Since large amounts of human resources are required to perform this knowledge extraction, it is desirable to automate as much of this process as possible.

Databases use attributes, A_i, and their associated values, a_{ij}, to represent information about quantities of interest. These attributes may represent both numeric (discrete and continuous) and nonnumeric quantities. A database is an organized set of these attributes and their values, a set of relations among these attributes, and a language for manipulating attributes and the relationships among them. This structure transforms raw data into information (18).

While information contained in a database may be accurate and complete, it is not knowledge. Using information as knowledge requires identification of the pertinent logical entailments hidden in that information. It is these logical entailments that allow inferences to be made from information contained in the database. Identification of logical entailments is complex and is usually done by

* This work was supported by the McDonnell Douglas Independent Research and Development program.

special algorithms (2). This paper is concerned with the automated extraction of knowledge from databases and the representation of this knowledge in a structure that can be processed by logical entailment algorithms (18).

The first step in performing knowledge extraction is to determine how knowledge will be represented. While various knowledge representation schemes have been developed for expressing concepts and their relationships to other knowledge, the relationships modeled by all these schemes can generally be expressed in terms of first-order logic expressions (2). One well-known knowledge-representation scheme is the rule-based system characterized by its knowledge base of facts and rules. In this paper, all references to rules apply to any knowledge representation which is used to model first-order logic expressions. The actual physical knowledge representation is secondary in importance.

The second step in performing knowledge extraction is to determine which algorithm should be used in the extraction process. One of the more popular and successful classes of algorithms which are used for this process are called decision-tree classifier systems. These systems take training instances as input and produce a set of rules as output. The rules output by the system are represented in the form of one or more decision trees (3,10,11,).

Recent research has focused on the automated extraction of knowledge from existing databases in an effort to reduce the development and maintenance costs of AI systems. This is a complex problem since concepts may take many forms, the identification of appropriate attributes is difficult, and sufficient information may not be available to support the formation of clear and accurate concepts. Other factors which contribute to the complexity of this problem are the difficulty in determining when extraction is complete and the difficulty of evaluating the knowledge produced.

The following sections describe an approach for extracting knowledge from databases which addresses many of these difficulties. The approach described is applicable in cases where the extracted knowledge can be represented as a set of rules. The extraction techniques use a class of inductive machine-learning techniques called decision-tree classifier systems. The section entitled Evaluation of Knowledge Extracted describes a metric which is useful for measuring the results of the extraction effort. The last section shows the results obtained by extracting knowledge from a "real" database (6).

EXTRACTION OF KNOWLEDGE FROM DATABASES

Type of Concepts to be Learned: One of the first steps in knowledge extraction is to determine the type of concepts to be learned. For instance, if one is trying to extract diagnostic information from a database, it is usually desirable to express the concepts being learned as rules. Machine learning techniques, such as decision-tree classifier systems, are proficient at this form of extraction. This approach is applicable to both numeric and nonnumeric data. However, continuous numeric-valued attributes present special problems (13).

Uncertainty plays a major role in knowledge extraction. Uncertainty involves both the uncertainty of facts and of rules. Fact uncertainty may be the result of noisy training examples. Noise is hard to identify since it is difficult to differentiate noise from "exceptions to a rule." Although several different approaches have been tried for handling noisy data (3,11), noise still presents a difficult problem for knowledge-extraction techniques. Rule uncertainty not only concerns the certainty with which conclusions can be asserted within a rule, but also the way in which uncertainties are propagated along rule chains. Both types of uncertainty introduce serious problems in knowledge extraction and continue to be active areas of research.

If the concepts to be learned are in the form of mathematical equations, standard operations research and statistical techniques such as regression (linear and nonlinear), correlation, and hypothesis testing may produce more satisfactory results (7). Although operations research and

statistical approaches are used primarily with numeric data, nonnumeric attributes can be assigned numeric values. Once the appropriate operations research techniques have been applied, the resulting equation(s) must then be interpreted in light of their nonnumeric counterparts (5).

The extraction techniques described below assume that the database on which extraction is to be performed consists of a set of records. For the purpose of knowledge extraction, each record is viewed as a set of attribute-value pairs along with one or more associated conclusions. Whether or not the database is a single entity or a distributed database is not important. Richardson (12) has developed an algorithm for combining information from numerous relations into single records. Each of these records becomes a training instance to the machine learning technique.

Representation of Concepts: Learned concepts will be represented in the form of rules. For example, the rule:

$$A_1(a_{11}) \wedge A_4(a_{43}) \wedge A_2(a_{22}) \Rightarrow c_6 \quad (1)$$

denotes the concept: if the value of attribute A_1 is a_{11}, the value of attribute A_4 is a_{43}, and the value of attribute A_2 is a_{22}, then c_6 may be concluded. (Boolean-valued expression $A_i(a_{ij})$ is true when a_{ij} is the value of A_i.) Fig. 1 shows how rule (1) would be represented by a decision tree.

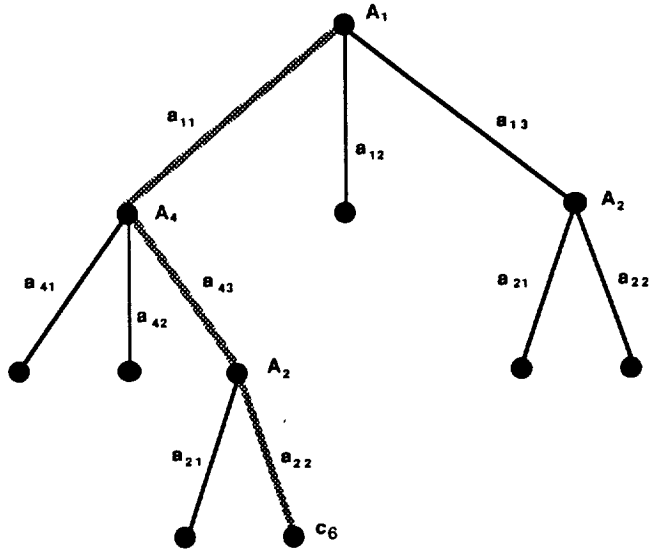

Fig. 1 Decision Tree Showing $A_1(a_{11}) \wedge A_4(a_{43}) \wedge A_2(a_{22}) \Rightarrow c_6$

Individual concepts are represented as paths in the decision tree. Each internal node represents an attribute A_i while each branch descending from A_i corresponds to a specific attribute value, a_{ij}. Each leaf node, c_j, represents a conclusion out of the set C of all possible conclusions.

Since more than one conclusion may exist at a leaf node, the concept shown in Fig. 1 will be represented by the tree shown in Fig. 2. The trees are identical with the exception of the label at the

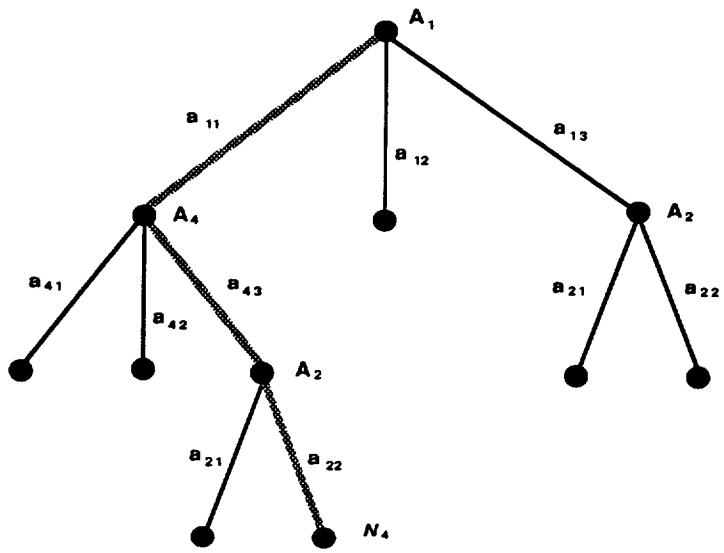

Fig. 2 Decision Tree Showing $A_1(a_{11}) \wedge A_4(a_{43}) \wedge A_2(a_{22}) \Rightarrow C(N4)$

leaf node. The label, N_4, represents the set of one or more conclusions, $c_j \in C$ occurring at this node. The expression:

$$A_1(a_{11}) \wedge A_4(a_{43}) \wedge A_2(a_{22}) \Rightarrow C(N_4)$$

denotes the rule where $C(N_4)$ is the conjunction of all $c_j \in N_4$. Note that: (1) multiple conclusions may be present at any leaf node, i.e., $|N_k| > 1$, and (2) any conclusion, c_j, may be present at more than one leaf node, i.e., $c_j \in N_k$ for more than one value of k. **Similar concepts** are concepts which have the same conclusion.

The problem of extracting knowledge in the form of decision trees reduces to the problem of constructing "correct" trees. Decision-tree classifier systems are a class of machine learning techniques which can be used to construct such trees.

DECISION-TREE CLASSIFIER SYSTEMS

Decision-tree classifier systems take training instances as input and output decision trees like that shown in Fig. 2 (3,10,11). These systems are called classifier systems because they separate input training instances into different classes. They are also referred to as induction systems since they induce knowledge from examples. Decision trees are frequently used to represent the results of this classification, hence the name decision-tree classifier systems.

During construction of the tree, the decision-tree classifier must determine the best attribute to be used to expand the tree at each node. It must also determine when no further attributes should be added to a path of the tree. Induction of decision trees may be incremental or nonincremental. In nonincremental induction, all training instances are processed at one time and the decision tree created. At this point, the learning process is considered completed. In incremental induction, learning is performed each time the decision tree is used to classify an instance. A well-known nonincremental induction technique called ID3 was developed by Quinlan (11) and is based on earlier work in induction by Hunt et al. (9). Two incremental versions of ID3 have been developed; ID4 by Schlimmer and Fisher (14) and ID5 by Utgoff (19).

Other types of machine learning approaches are applicable to knowledge extraction. These include case-based reasoning(15), explanation-based learning(4), and genetic algorithms(8). They will not be discussed in this paper.

DECIDING WHAT TO EXTRACT

The selection of the appropriate database attributes to participate in the extraction process is critical both to the quality of the knowledge extracted and to the efficiency of the extraction process. The choice of attributes depends on the type of concepts being learned. In many cases, the domain expert may be able to provide advice on which attributes are likely to be important. The attributes chosen to participate in the extraction process make up what is called the description space, viz;

$$D = \{ A_1, A_2, ..., A_n \}.$$

In an effort to keep the description space as small as possible, statistical and mathematical programming techniques such as regression analysis and correlation can be used to help identify database attributes which are dependent on each other. When selecting database attributes to be included in the description space, it is seldom necessary to include attributes which are dependent on others already in the set. Mathematical programming techniques can be used to help identify linear and some types of nonlinear dependence among attributes. In some cases, simple plots of database values may help identify appropriate attributes.

Once a candidate description space is identified, the next step is to perform knowledge extraction using only a subset of the training instances available. This is desirable since the machine learning mechanism being used may also help identify relationships among attributes which have not been detected by earlier efforts. After evaluating these initial results, it may be possible to further revise/refine the description space.

All of these efforts are designed to keep the complexity of the extraction process to a minimum. Minimization of complexity is desirable because a database may contain a large number of attributes. Simply using all database attributes in the knowledge extraction process would only increase the complexity of the extraction process without adding additional knowledge.

EVALUATION OF KNOWLEDGE EXTRACTED

Since the accuracy of the concepts learned as well as the complexity of the tree constructed is determined by both the quality and quantity of training instances and by the way the classifier system chooses attributes for the tree, it is desirable to evaluate the "quality" of the knowledge extracted. Knowledge quality can be measured in different ways, including the correctness and thoroughness of the knowledge extracted and the certainty with which the knowledge structure can differentiate between the concepts learned.

Evaluating knowledge correctness is necessary to determine how well the concepts learned compare with what is known about the "real" world. Correctness evaluations are done in a manner similar to verification and validation of expert systems (17). A common set of test suites is evaluated first by using the extracted rules and then by domain experts. Next, these test results are compared. This approach helps verify that the set of concepts learned is consistent with domain experts' knowledge. Failure to adequately satisfy correctness tests may be the result of poor attribute selection, poor extraction techniques, or an inadequate number of training instances.

In many cases, domain experts discover that the knowledge extracted is correct but not thorough. This is evidenced by the fact that "pieces" of knowledge are found missing during the tests for correctness. This may indicate an inadequate number of training instances in the database. In these cases, additional knowledge may have to be added by domain experts.

To measure how well a decision tree differentiates between concepts, the authors developed an approach for evaluating the quality of a learned decision tree by measuring certain characteristics of the tree. This approach complements the work of domain experts and is especially useful in cases where multiple conclusions exist at a leaf node, i.e. $|N_k| > 1$, for some value(s) of k. The approach uses the Concept Strength Metric (CSM) described below.

The development of the Concept Strength Metric was motivated by the need to construct diagnostic advisors for use in aircraft maintenance (1,16). By utilizing inductive learning systems, it is possible to construct diagnostic advisors which can assist in maintaining their own knowledge bases. However, one of the problems arising from such learning systems concerns the quality of concept differentiation since it is rarely the case that all concepts will be learned perfectly.

The Concept Strength Metric value, $E(c_j)$, for each conclusion c_j, is the weighted measurement indicating that, given the decision-tree's current level of experience, it can uniquely differentiate conclusion c_j. The value $E(c_j)$ is the sum of individual weighted strengths, $E_k(c_j)$, for c_j at each leaf node, viz:

$$E(c_j) = \sum_{k=1}^{\lambda} E_k(c_j),$$

where λ is the number of leaf nodes in the tree.

The weighted strength, $E_k(c_j)$, for conclusion c_j at node N_k, is the weighted probability that conclusion c_j can be clearly differentiated by the path leading to node N_k. It is calculated by computing:

$$E_k(c_j) = \frac{\delta_{jk}}{\sum_{i=1}^{|C|} \delta_{ik}} * \frac{\delta_{jk}}{\sum_{h=1}^{\lambda} \delta_{jh}} = \alpha_{jk}\beta_{jk}$$

where δ_{jk} denotes the number of times conclusion c_j has appeared at node N_k and $|C|$ denotes the number of possible conclusions in the tree. The frequency of c_j occurrences in the tree is given by η_j. The frequency of all conclusions occurring at node N_k is ξ_k. Note that:

$$\xi_k = \sum_{i=1}^{|C|} \delta_{ik} \quad \text{and} \quad \eta_j = \sum_{k=1}^{\lambda} \delta_{jk}.$$

The factors

$$\alpha_{jk} = \frac{\delta_{jk}}{\sum_{i=1}^{\chi} \delta_{ik}} = \frac{\delta_{jk}}{\xi_k} \text{ and } \beta_{jk} = \frac{\delta_{jk}}{\sum_{h=1}^{\lambda} \delta_{jh}} = \frac{\delta_{jk}}{\eta_j}$$

are of interest. The first, α_{jk}, denotes the fraction of all conclusions at node N_k which are c_j. The

larger the value of δ_{jk}, the higher the likelihood of uniquely identifying conclusion c_j at this node. Since

$$\sum_{j=1}^{|C|} \alpha_{jk} = 1$$

the larger the number of conclusions at node N_k, the smaller the likelihood that all conclusions will be uniquely differentiated at this node. If only one conclusion, say c_1, is present at node N_k, then $\alpha_{1k} = 1$. If $c_j \notin N_k$, then $\alpha_{jk} = 0$.

The factor β_{jk} scales α_{jk} by the fraction of c_j occurrences at all leaf nodes. This factor scales the current knowledge about conclusion c_j at node N_k with respect to all the information about conclusion c_j. Hence, the product, $E_k(c_j) = \alpha_{jk}\ \beta_{jk}$, is the weighted measure that conclusion c_j can be clearly differentiated by the path leading to node N_k, given the current level of knowledge.

To illustrate the Concept Strength Metric, consider the tree shown in Fig. 3. The values at each leaf node N_k indicate the nonzero number of times each conclusion $c_j \in C$ has occurred at that node. For example, at node N_1, conclusion c_3 occurred twelve times. No other conclusions occurred at this node. This experiment contained 168 training instances with $|C| = 5$, $\eta_1 = 65$, $\eta_2 = 50$, $\eta_3 = 20$, $\eta_4 = 23$, $\eta_5 = 10$, $\xi_1 = 12$, $\xi_2 = 75$, $\xi_3 = 60$, $\xi_4 = 13$, and $\xi_5 = 8$. Table I shows the values of $E_k(c_j)$ and $E(c_j)$ for each conclusion.

Several interesting results are observable from the table. Each $E_k(c_j)$ in the table has a value between zero and one. Each $E_k(c_j)$ value represents the weighted probability that conclusion c_j can be uniquely identified when tree traversal leads to N_k. Conclusion c_3 is of particular interest since it appears at two nodes and it does not appear with any other conclusions. Hence, based on the present knowledge level of the tree, it is possible to positively identify all similar concepts whose

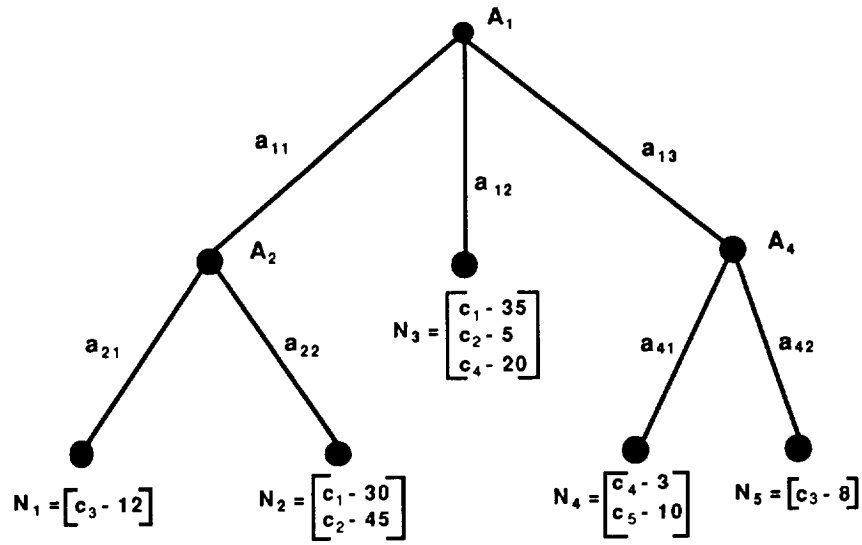

Fig. 3 Sample Tree Showing Conclusions at Each Node

k	$E_k(c_1)$	$E_k(c_2)$	$E_k(c_3)$	$E_k(c_4)$	$E_k(c_5)$
1	0	0	0.60	0	0
2	0.18	0.54	0	0	0
3	0.31	0.0083	0	0.29	0
4	0	0	0	0.03	0.77
5	0	0	0.40	0	0
$E(c_j)$	0.49	0.548	1	0.32	0.77

Table I. CSM Values for Fig. 3

conclusion is c_3. This is reflected by the fact that $E(c_3) = 1$. It can be shown mathematically that $E(c_j) = 1$ whenever for each node N_k such that $c_j \in N_k$, then c_j is the only conclusion at N_k. Such conclusions are said to be **completely differentiated** by the learned decision tree.

The next most completely differentiated set of similar concepts are those whose conclusion is c_5 since $E(c_5) = 0.77$. Conclusions c_1, c_2, and c_4 are not as completely differentiated as c_3 and c_5. These lower CSM values may imply that multiple concepts are present at several leaf nodes. Individual values of $E_k(c_j)$ provide additional information about each conclusion. For example, the fact that $E_3(c_2) = 0.0083$. indicates that this decision tree is a poor differentiator of c_2 at node three. The cause of this poor differentiation may be the result of noise or it may be due to a lack of training examples which contain this concept.

The $E(c_j)$ and $E_k(c_j)$ values can be used to determine when the decision tree has reached the desired level of concept differentiation. They can also be used to guide the learning process by indicating what types of additional knowledge are needed to improve the tree.

KNOWLEDGE EXTRACTION EXPERIMENT

The objective of this experiment was to extract rules from a database of iris flowers compiled by Fisher (6). The database contained four flower-description attributes with an associated flower type: virginica, versicolor, or setosa. Initial plots of attribute vs iris type suggested that neither sepal length nor sepal width alone were sufficient to predict iris type since several values for each of these attributes were associated with multiple iris types . Hence, these two attributes were chosen for the description space.

Utgoff's ID5 (19) was chosen as the extraction mechanism. ID5 was extended to calculate values of the CSM. Results from these experiments are shown in Table II. Given there were two attributes; sepal width with 26 values and sepal length with 41 values; and there were 94 leaf nodes, it can be seen that the resulting decision tree was wide and shallow. There were ten leaf nodes which contained multiple conclusions. At the conclusion of the learning process, the training set was used to test how accurately iris variety could be predicted. The tree produced was able to distinctly classify only 84% of the training instances. This means that 16% of the training instances fell into one of the ten leaf nodes containing multiple conclusions.

Evaluation of the knowledge extracted was based primarily on how well the decision tree differentiated between concepts. The CSM values shown in Table II indicate that the CSM value of clearly differentiating virginica is the same as the CSM value for versicolor. The fact that E(setosa) = 1.0 implies that the tree clearly distinguishes the setosa species, since all leaf nodes containing setosa as a conclusion do not contain any other conclusions.

# Training Instances	150
# Training Attributes	2
# Internal Nodes	20
# Leaf Nodes	94
# Leaf Nodes containing multiple conclusions	10
E(virginica)	= 0.88
E(versicolor)	= 0.88
E(setosa)	= 1.00

Table II. Results of Iris Tests

CONCLUSIONS

This paper has described the general process of extracting knowledge from databases using decision-tree classifier systems. These learning mechanisms, based on induction, extract knowledge from input training instances and represent it in the form of decision trees. The Concept Strength Metric (CSM) was described for measuring the amount of concept differentiation in these decision trees. This result is important in helping to determine when sufficient knowledge extraction has been performed. By examining values of $E_k(c_j)$ and $E(c_j)$, it can be decided which conclusions require additional training instances to improve concept differentiation. The CSM may be effectively used to evaluate concept differentiation in any decision tree. Experimental results using the Concept Strength Metric are generating interest among practitioners in the diagnostic community.

REFERENCES

1. Bond, W. E., St. Clair, D. C., Flachsbart, B. B., and Vigland, A. R., *Integration of an Adaptive Diagnostic Expert System into an Avionics Test Environment*, **Proceedings of the Third Annual Expert Systems in Government Conference**, IEEE Computer Society, October 1987, pp. 132-136.

2. Brachman, R.J., *The Basics of Knowledge Representation*, **AT&T Technical Journal**, Vol. 67, No. 1, 1988, pp. 7-24.

3. Clark, P. and Niblett, T., *The CN2 Induction Algorithm*, **Machine Learning**, Vol 3, No. 4, 1989, pp. 261-284.

4. DeJong, K. and Mooney, R.J., *Explanation-based Learning: An Alternative View*, **Machine Learning**, Volume 1, 1986, pp. 145-176.

5. Draper, N. R., Smith, H., **Applied Regression Analysis**, 2nd ed. New York : Wiley, 1981.

6. Fisher, R. A., *The Use of Multiple Measurements in Taxonomic Problems*, **Annals of Eugenics**, Vol. 7, 1936.

7. Hillier, F. S. and Lieberman, G. J., **Introduction to Operations Research**, 4th ed., San Francisco : Holden-Day, 1986.

8. Holland, J.H., *Properties of the Bucket Brigade Algorithm*, **Proceedings of the First International Conference on Genetic Algorithms and Their Applications**, Pittsburgh, PA: Lawrence Erlbaum, 1985, pp. 1-7.

9. Hunt, E. B., Martin, J., and Stone, P. J., **Experiments in Induction**, Academic Press, 1966.

10. Michalski, R. S., Mozetic, I., Hong, J., and Lavrac, N., *The Multipurpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains*. **Proceedings of the Fifth National Conference on Artificial Intelligence**, Morgan Kaufmann, 1986, pp. 1041-1045.

11. Quinlan, J.R., *Induction of Decision Trees*, **Machine Learning**, Vol. 1, 1986, pp. 81-106.

12. Richardson, J. M., **Deduction of a Functional Dependency From a Set of Functional Dependencies**, M.S. Thesis, University of Missouri -- Rolla, 1988.

13. Sabharwal, C. L., St. Clair, D. C., Hacke, K., and Bond, W. E., *Representation of Continuous Attributes in IDx Classifier Systems*, **Proceedings of the Fourth International Symposium on Methodologies for Intelligent Systems: Poster Session Program**, Oak Ridge National Laboratory, ORNL/DSRD-24, October 1989, pp. 167-176.

14. Schlimmer, J. C., and Fisher, D., *A Case Study of Incremental Concept Induction*, **Proceedings of the Fifth National Conference on Artificial Intelligence**, Vol. 1, Morgan Kauffman Publishers, Inc., August 1986 , pp. 496-501.

15. Slade, S., *Case-based Reasoning: A Research Paradigm*, Yale Department of Computer Science, Report # YALEU/CSD/RR #644, August 1988.

16. St. Clair, D. C., Bond, W. E., Flachsbart, B. B., and Vigland, A. R., *An Architecture for Adaptive Learning in Rule-Based Diagnostic Expert Systems*, **AI in Armament Workshop -- Diagnostics**, American Institute of Aeronautics & Astronautics, March 1988. (Reprinted from **1987 Proceedings of the Fall Joint Computer Conference**, IEEE Computer Society, October 1987, pp. 678-687.)

17. St. Clair, D. C., Bond, W. E., and Flachsbart, B. B., *Using Output to Evaluate and Refine Rules in Rule-Based Expert Systems*, **Proceedings of the Third Conference on Artificial Intelligence for Space Applications, Part I**," NASA Conference Publication 2492, November 1987, pp. 9-14.

18. Ullman, J. D., **Principles of Database and Knowledge-base Systems**, Computer Science Press, Vol. 1, 1988, pp. 23-25..

19. Utgoff, P. E., *ID5: An Incremental ID3*, **Proceedings of the Fifth International Conference on Machine Learning**, Morgan Kaufmann Publishers, Inc., June 1988, pp. 107-120.